PATENT ABSTRACTS OF JAPAN

(11)Publication number : 06-103091
(43)Date of publication of application : 15.04.1994

(51)Int.Cl. G06F 9/46

(21)Application number : 04-249830
(22)Date of filing : 18.09.1992

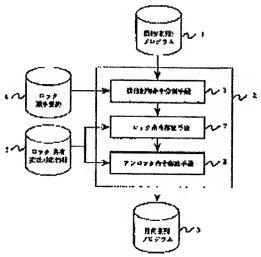
(71)Applicant : HITACHI LTD
(72)Inventor : TSUNETOMI KUNIHIKO
YAMAGUCHI SHINICHIRO
KAMIWAKI TADASHI

(54) METHOD AND DEVICE PARALLEL EXCLUSIVE CONTROL INSTRUCTION TRANSLATION AND DATA PROCESSOR

(57)Abstract:

PURPOSE: To evade the generation of a deadlock in parallel programs and increase the parallelism of the programs.

CONSTITUTION: A lock order code storage means 4 which stores the code of lock order, a lock-common variable correspondence information storage means 5 which stores the correspondence between common variables and lock variables, an exclusive control instruction dividing means 6 which performs division into plural lock instructions and unlock instructions according to the order code storage means 4, a lock instruction moving means 7 which moves the lock instruction to behind a program 1 by referring to the lock-common variable correspondence information storage means 5, and an unlock instruction moving means 8 which moves the unlock instructions to before a program 2 by referring to the lock-common variable correspondence information storage means 5 are provided. The lock instruction moving means 7 moves the lock instructions to behind the program while following the lock order code and the unlock instruction moving means 8 moves the unlock instructions to before the program to narrow down a critical section, thereby increasing the parallelism of the program.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開平6－103091
(43)公開日　平成６年(1994)４月15日

(51)Int.Cl.[5]	識別記号	庁内整理番号	Ｆ Ｉ	技術表示箇所
Ｇ０６Ｆ　9/46	３４０	Ｇ　8120－5Ｂ		

審査請求　未請求　請求項の数６(全 12 頁)

(21)出願番号　特願平4－249830

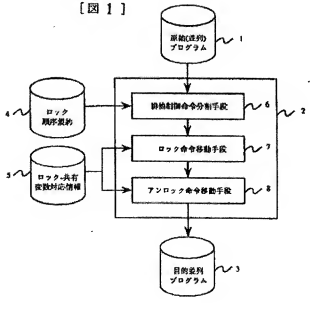
(22)出願日　平成４年(1992)９月18日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地
(72)発明者　恒富　邦彦
茨城県日立市久慈町4026番地　株式会社日立製作所日立研究所内
(72)発明者　山口　伸一朗
茨城県日立市久慈町4026番地　株式会社日立製作所日立研究所内
(72)発明者　上脇　正
茨城県日立市久慈町4026番地　株式会社日立製作所日立研究所内
(74)代理人　弁理士　秋本　正実

(54)【発明の名称】　並列排他制御命令翻訳方法及びその装置とデータ処理装置

(57)【要約】
【目的】　並列プログラムでデッドロックの発生を回避し、且つプログラムの並列性を高める。
【構成】　ロック順序の規約を記憶しておくロック順序規約記憶手段４と、共有変数とロック変数との対応を記憶しておくロック－共有変数対応情報記憶手段５と、順序規約記憶手段４に基づいて複数のロック命令，アンロック命令に分割する排他制御命令分割手段６と、ロック－共有変数対応情報記憶手段５を参照してロック命令をプログラム1の後方に移動するロック命令移動手段７と、ロック－共有変数対応情報記憶手段５を参照してアンロック命令をプログラム２の前方に移動するアンロック命令移動手段８を設ける。ロック命令移動手段７がロック命令をロック順序規約を遵守しながらプログラムの後方に移動し、アンロック命令移動手段８がアンロック命令をプログラムの前方に移動し、クリティカルセクションを狭めることにより、プログラムの並列性を上昇させる。

［図１］



1

【特許請求の範囲】

【請求項１】　複数のプログラム間で共有する複数の共有変数の排他制御を行うために、該共有変数に対するロック変数を持ち、該ロック変数に対してロック命令を発行することによって、該共有変数を1つのプログラムが排他的に使用する状態を作り、該ロック変数に対して、アンロック命令を発行することによって、該共有変数を１つのプログラムが排他的に使用する状態を解除して、排他制御を行う並列処理において、複数のロック変数に対するロックを同時に指定する統合ロック命令と、複数のロック変数に対するアンロックを同時に指定する統合アンロック命令と、該統合ロック命令と統合アンロック命令を、各ロック変数に対するロック命令とアンロック命令に分解し、ロック命令とアンロック命令をプログラム中で移動して、クリティカルセクションを狭めるロックデコード手段を有することを特徴とする排他制御命令翻訳処理装置。

【請求項２】　請求項１において、ロックデコード手段は、ロック順序の規約を記憶しておくロック順序規約記憶手段と、共有変数とロック変数との対応を記憶しておくロック一共有変数対応情報記憶手段と、前記ロック順序規約記憶手段に基づいて統合ロック命令，アンロック命令を各ロック変数に対するロック命令，アンロック命令に分割する排他制御命令分割手段と、前記分割された複数のロック命令を前記ロック一共有変数対応情報記憶手段を参照してロック変数と共有変数との関係からプログラム中で再配置する命令移動手段とを備えることを特徴とする排他制御命令翻訳装置。

【請求項３】　請求項１において、ロックデコード手段は、ロック順序の規約を記憶しておくロック順序規約記憶手段と，共有変数とロック変数との対応を記憶しておくロック一共有変数対応情報記憶手段と、前記ロック順序規約記憶手段に基づいて統合ロック命令，アンロック命令を各ロック変数に対するロック命令，アンロック命令に分割する排他制御命令分割手段と、ロック一共有変数対応情報記憶手段を参照してロック順序を遵守し且つロックしようとするロック変数に対応する共有変数へのアクセス命令に最も近い場所にロック命令を移動するロック命令移動手段と、ロック一共有変数対応情報記憶手段を参照しアンロックしようとするロック変数に対応する共有変数へのアクセス命令に最も近い場所にアンロック命令を移動するアンロック命令移動手段とを備えたことを特徴とする排他制御命令翻訳装置。

【請求項４】　請求項２において、ロック一共有変数対応情報記憶手段は、共有変数のシンボルとロック変数のシンボルと次のロック一共有変数対応情報記憶手段へのポインタより成り、ロック規約記憶手段は、該ロック共有変数対応情報記憶手段をロック順序の早い変数の順に並べてロック規約を記憶することを特徴とする排他制御命令翻訳装置。

2

【請求項５】　複数のプログラム間で共有する複数の共有変数の排他制御を行うために、該共有変数に対するロック変数を持ち、該ロック変数に対してロック命令を発行することによって、該共有変数を1つのプログラムが排他的に使用する状態を作り、該ロック変数に対して、アンロック命令を発行することによって、該共有変数を１つのプログラムが排他的に使用する状態を解除して、排他制御を行う並列処理において、複数のロック変数に対するロックを統合ロック命令で同時に指定し、複数のロック変数に対するアンロックを統合アンロック命令で同時に指定し、前記統合ロック命令と統合アンロック命令を、各ロック変数に対するロック命令とアンロック命令に分解し、ロック命令とアンロック命令をプログラム中で移動して、クリティカルセクションを狭めることを特徴とする排他制御命令翻訳処理方法。

【請求項６】　複数のプログラム間で共有する複数の共有変数の排他制御を行うために、該共有変数に対するロック変数を持ち、該ロック変数に対してロック命令を発行することによって、該共有変数を1つのプログラムが排他的に使用する状態を作り、該ロック変数に対して、アンロック命令を発行することによって、該共有変数を１つのプログラムが排他的に使用する状態を解除して、排他制御を行う並列処理のデータ処理装置において、複数のロック変数に対するロックを同時に指定する統合ロック命令と、複数のロック変数に対するアンロックを同時に指定する統合アンロック命令と、該統合ロック命令と統合アンロック命令を、各ロック変数に対するロック命令とアンロック命令に分解し、ロック命令とアンロック命令をプログラム中で移動して、クリティカルセクションを狭めるロックデコード手段を有することを特徴とするデータ処理装置。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は≪並列プログラム中の排他制御命令記述の間違いによるデッドロックを回避し、プログラムの排他制御部分を小さくするよう排他制御命令を自動的に移動してプログラムの並列性を上昇させる並列排他制御命令翻訳方法及びその装置とデータ処理装置に関する。

【０００２】

【従来の技術】２つ以上のプログラムが並列に実行する場合、共有に使用する共有変数等の資源に対して処理を同時に行うと、矛盾した動作をする可能性がある。このため共有変数について処理を行うときは、他のプログラムが当該共有変数を扱えないように、排他制御（共有変数に対応するロック変数へのロック，アンロック）を行う。

【０００３】しかし、複数のロック変数をロックする必要があるプログラムが、並行に動作している場合には問題がある。各プログラムがロックしているロック変数

を、お互いにロックしようとすると、プログラムがロック待ちで永久に停止してしまう状態（デッドロック）に陥る。例を、図6（a）に示す。尚、図6，図7において、変数a、b、cは並列プログラムAと並列プログラムBの間での共有変数、lock_a、lock_b、lock_cは共有変数a、b、cに対応するロック変数である。このとき、並列プログラム130でロック命令1303を実行すると同時に、並列プログラム140でロック命令1402を実行した場合にデッドロックとなる。

【0004】従来、この問題を解決するために、ロック変数のロック順序をあらかじめ決定しておき、プログラマがロック命令の順序を規約に間違えないように注意してプログラムに挿入する必要があった。また、特開平3－224036号公報記載の従来技術では、全てのロック変数のロック順序を予め記憶しておき、一ロック命令で、ロック順序にしたがって全ロック変数をロックしている。

【0005】

【発明が解決しようとする課題】ロック命令を逐一プログラマが記述する方法は、プログラマの負担が大きい。また、コーディング中にロック順序規約を変更する必要が発生した場合、ロック命令の位置を大きく書き直さなくてはならない。特開平3－224036号公報記載の従来技術は、上記の問題を解決している。しかし、ロック命令とアンロック命令で挟まれた排他制御される領域（クリティカルセクション）の共有変数のために、必要のないロック変数のロックも行ってしまうという問題がある。また、複数のロック変数のロック命令との間に、処理を記述することができないので、プログラムのクリティカルセクションを大きくし、並列性を制限するという問題がある。例を、図6（b）に示す。この場合、並列プログラム150の命令1502は、lock_aとlock_bさえロックすれば実行できる。並列プログラム160の命令1604は、lock_cさえロックすれば実行できる。したがって、命令1502と命令1604は並列処理可能である。しかし、上述した従来例の手法では、ロック変数の全てをロックしてクリティカルセクションを実行するので、並列に処理が実行されず、並列性がよくないという問題がある。

【0006】本発明の目的は、簡潔なロック操作記述を用い、クリティカルセクション内で必要なロック操作のみを決められた順序にしたがって実行する並列排他制御命令翻訳方法及びその装置並びにデータ処理装置を提供することにある。

【0007】本発明の他の目的は、個々のロック変数に対するロック命令，アンロック命令をロック変数に対応する共有変数へのアクセス命令に最も近く且つロック順序を遵守する位置に移動することによりクリティカルセクションを小さくしてプログラムの並列性を高める並列排他制御命令翻訳方法及びその装置とデータ処理装置を提供することにある。

【0008】

【課題を解決するための手段】上記目的は、複数のロック変数に対するロック命令，アンロック命令を、一命令のロック命令，アンロック命令でまとめて記述されたプログラムを翻訳し、実行する計算機システムにおいて、ロック順序の規約を記憶しておくロック順序規約記憶手段と、共有変数とロック変数との対応を記憶しておくロッカー共有変数対応情報記憶手段と、前記ロック順序規約記憶手段とロック命令の引数のロック変数に基づいて一命令のロック命令，アンロック命令を複数のロック命令列，アンロック命令列に分割する排他制御命令分割手段と、ロッカー共有変数対応情報記憶手段を参照しロック順序を違守し且つロック変数に対応する共有変数へのアクセス命令に最も近い場所にロック命令を移動するロック命令移動手段と、ロッカー共有変数対応情報記憶手段を参照しアンロックしようとするロック変数に対応する共有変数へのアクセス命令に最も近い場所にアンロック命令を移動するアンロック命令移動手段とを設けることで、達成される。

【0009】【作用】排他制御命令分割手段が、ロック順序規約記憶手段が予め記録するロック順序の規約を参照し、規約どおりのロック命令列に展開することにより、デットロックを回避する。また、共有変数とロック変数との対応を記録したロッカー共有変数対応情報記憶手段を参照して、ロック命令移動手段が、展開されたロック命令をプログラムの後方に移動し、アンロック命令移動手段が、展開されたアンロック命令をプログラムの前方に移動することにより、クリティカルセクションを狭め、プログラムの並列性を上昇させる。

【0010】

【実施例】以下、本発明の実施例について図面を参照して説明する。図1は、本発明の一実施例に係る並列排他制御命令翻訳装置であるコンパイラのブロック図である。図1において、1は原始プログラム、2はコンパイラ、3は目的並列プログラム、4はロック順序規約記憶手段、5はロッカー共有変数対応情報記憶手段である。コンパイラ2は排他制御命令分割手段6、ロック命令移動手段7、アンロック命令移動手段8を備える。

【0011】図2（a）は、ロック命令移動手段7と、アンロック命令移動手段8が使用するロッカー共有変数対応情報テーブルであり、ロック変数と共有変数との対応を記憶するのである。テーブルは共有変数ごとに設けられ、共有変数5aと対応するロック変数5b、次ロッカー共有変数対応情報テーブルのアドレス5cから構成される。

【0012】図2（b）は、ロック順序規約記憶手段の構成図である。同図に示すように、ロック共有対応情報テーブルの連結順序によって、ロック順序を記憶する。図2（c）は、共有変数のプログラム中での宣言説明図である。共有変数宣言sharedにより定義された共有変数

は、対応するロック変数が確保され、これと同時にロック一共有データ対応情報テーブルに記入されて、ロック順序規約記憶手段に連結される。

【００１３】以下、図７の様に、ユーザが記述した原始プログラム１１０ａ，１２０ａのロック命令が、引数としてロック変数をもつ場合について述べる。

【００１４】図３は、排他制御命令分割手段６の処理手順の一例を示すフローチャートである。以下、このフローチャートに従って説明する。排他制御命令分割手段６は、先ず、原始プログラムの最初から走査してゆき、ロック命令を検索する（ステップ601）。ロック命令が存在すると（ステップ601のNo）、これを分割処理対象のロック命令Lとする（ステップ602）。

【００１５】次に、ロック命令Lの全ロック変数について、ロック順序規約のロック順序情報を調べたか否かを調べる（ステップ603）。ロック変数のうちロック順序が調べられていないロック変数Lockがある場合（ステップ603のNo）、Lockのロック順序を、ロック規約情報４から調べ、テンポラリ領域にLockとLockのロック順序を書き出す（ステップ605）。その後ステップ603に戻り、ロック命令Lの全てロック変数とそのロック順序がテンポラリ領域に書き出されるまで、繰り返す。

【００１６】ロック命令Lのロック変数のロック順序が全て調べられた場合（ステップ603のYes）、ステップ605で記述されたテンポラリ領域を参照し、ロック命令Lを、ロック順序の早い順に並ぶロック変数毎のロック命令列に置き換える（ステップ606）。これにより、ロック命令Lの分解処理が終了する。

【００１７】その後、原始プログラムをロック命令Lの位置から後方に走査し、Lに対応するアンロック命令Uを検索する（ステップ607）。そして、テンポラリ領域を参照し、Uを、ロック順序の遅い順に並ぶロック変数毎のアンロック命令列に置き換える（ステップ608）。これにより、アンロック命令Uが分割される。その後、ステップ601に戻り、原始プログラムの全てのロック命令を分割するまで繰り返す。ロック命令を全て分解すると（ステップ601のYes）、終了する。

【００１８】図４は、ロック命令移動手段の処理手順に一例を示すフローチャートである。以下、このフローチャートに従って説明する。ロック命令移動手段７は、先ず、プログラムの最後から走査してゆき、ロック命令を検索する（ステップ701）。ロック命令が存在すると（ステップ701のNo）、これを検査対象のロック命令Lとする（ステップ702）。

【００１９】次に、ロック命令Lから後ろの命令を順に検索する（ステップ703）。現在検索された命令を命令Iとする。命令Iが、ロック命令Lのロック変数Lockを引数とするアンロック命令であるかを調べる（ステップ704）。もし、命令IがLockを引数とするアンロック命令である場合（ステップ704のYes）、ロック命令L、アンロック命令Iをプログラムから削除し（ステップ708）、ステップ701に戻る。ユーザのロック変数の指定に余分なロック変数がある場合、この動作が起こる。

【００２０】また、命令IがLockを引数とするアンロック命令でない場合（ステップ704のNo）、命令Iが、ロック命令であるかを調べる（ステップ705）。命令Iがロック命令である場合（ステップ705のYes）、命令Iの直前にロック命令Lを移動する（ステップ707）。これにより、ロック順序の早いロック変数のロックが、ロック順序の遅いロックを追い越してしまうことを防ぐ。

【００２１】命令Iがロック命令でない場合（ステップ705のNo）、ロック一共有変数対応情報テーブル５を参照し、命令Iがアクセスする変数のうち、Lockに対応する共有変数があるかを調べる（ステップ706）。命令Iが、Lockに対応する共有変数にアクセスしていない場合（ステップ706のNo）、ステップ703に戻り、命令Iの次の命令を、新たに命令Iとして処理を繰り返す。これにより、ロック命令Lが移動可能な、プログラムの最後方の位置の命令Iを検出する。

【００２２】命令Iが、Lockに対応する共有変数にアクセスしている場合（ステップ706のYes）、命令Iの直前にロック命令Lを移動する。その後ステップ701に戻り、ロック命令Lよりプログラムの前方にあるロック命令を、新たなロック命令Lとし、移動処理を繰り返す。全てのロック命令について移動処理が終了した場合（ステップ701のYes）、ロック命令移動手段7の処理を終了する。

【００２３】図７のプログラム１１０ｂ，１２０ｂは、上述した排他制御命令分割手段６の処理結果である。ただし、図７の並列プログラムでは、ロック順序規約はlock_a→lock_b→lock_cの順にロック変数をロックすると決められているとする。並列プログラム１１０ａ，１２０ａにおけるロック命令1110，1210、アンロック命令1140、1250が、ロック順に並ぶ、ロック変数毎のロック命令列1111～1113、1211～1212、アンロック命令列1141～1142，1251～1252になる。このプログラム１１０ｂ，１２０ｂが、ロック命令移動手段の入力プログラムとなる。

【００２４】図５は、アンロック命令移動手段８の処理手順の一例に係るフローチャートである。以下、このフローチャートに従って説明する。アンロック命令移動手段８は、まず、プログラムの最初から走査してゆき、アンロック命令を検索する（ステップ801）。アンロック命令が存在すると（ステップ801のNo）、これを検査対象のアンロック命令Uとする（ステップ802）。

【００２５】次に、アンロック命令Uから前の命令を順に選択する（ステップ803）。現在選択されている命令を命令Iとする。そして、ロック一共有変数対応情報テーブル５を参照し、命令Iがアクセスする変数のうち、アンロック命令のロック変数Lockに対応する共有変数が

あるかを調べる（ステップ804）。命令Iが、Lockに対応する共有変数にアクセスしていない場合（ステップ804のNo）、ステップ803に戻り、命令Iの次の命令を、新たに命令Iとして処理を繰り返す。これにより、アンロック命令Uが移動可能な、プログラムの最前方の位置の命令Iを検出する。

【００２６】命令Iが、Lockに対応する共有変数にアクセスしている場合（ステップ804のYes）、命令Iの直前にアンロック命令Uを移動する。その後ステップ801に戻り、アンロック命令Uよりプログラムの後方にあるアンロック命令を、新たなアンロック命令Uとし、移動処理を繰り返す。全てのアンロック命令について移動処理が終了した場合（ステップ801のYes）、アンロック命令移動手段8の処理を終了する。

【００２７】図７のプログラム１１０ｃ，１２０ｃは、並列プログラム１１０ｂ，１２０ｂをロック命令移動手段７、アンロック命令移動手段８で処理結果である。プログラム110cでは、ロック命令1133のロック変数が命令1120のアクセスする共有変数とは関係がなく、命令1120がロック命令でないので、ロック命令1133は命令1120の後方に移動されている。また、プログラム120cでは、アンロック命令1251のロック変数が命令1240のアクセスする共有変数と関係がないため、アンロック命令1251が命令1240の前に移動されている。

【００２８】次に本発明の第２実施例について説明する。前述した実施例１は、図７に示すように、ユーザが記述した原始プログラムのロック命令が、引数としてロック変数をもつ場合である。第２実施例として、ロック命令が引数を持たない記述をした場合について説明する。

【００２９】第１実施例の場合、図３のロック分割手段６の処理において、ロック命令の引数のロック変数のみ、ロック順序をテンポラリ領域に書き出した（ステップ603，604，605）。これを、ロック順序規約に記録された全ロックのロック順序をテンポラリ領域に記述するよう変更する。これにより、ロック命令Lは、ロック順序規約の順序に並ぶ全ロック変数毎のロック命令列に置き換えられる（ステップ606）。アンロック命令Uは、ロック順序規約の逆の順序に並ぶ全ロック変数毎のアンロック命令列に置き換えられる（ステップ607）。

【００３０】上記処理の出力プログラムは、排他制御される領域の共有変数とは関係ないロック命令，アンロック命令が含まれる。しかし、このロック命令，アンロック命令は、図４のロック命令移動手段7において検出され（ステップ704）、削除される（ステップ708）。そして、第１実施例と同一の目的プログラムである図７のプログラム110c，120cが得られる。

【００３１】上述した実施例によれば、複数のロック変数をまとめて指定する簡単な記述のロック命令が，予め記憶しておいた順序のロック命令に自動的に展開されるので、プログラムのコーディングが容易となると同時に、ロック順序の間違いによるデッドロックをなくす効果がある。また、展開されたロック命令，アンロック命令は、これに挟まれたクリティカルセクションが小さくなるように自動的に移動されるので、プログラムの並列性が向上する効果がある。すなわち、図６（ｂ）のプログラム150，160において、並列処理不可能であった命令1502，1604が、図７のプログラム110c，120cでは並列処理可能となる。

【００３２】

【発明の効果】本発明によれば、並列プログラムでデッドロックの発生を回避でき、しかもプログラムの並列性を高めることができる。

【図面の簡単な説明】

【図１】本発明の一実施例に係る排他制御命令翻訳装置の構成図である。

【図２】図１のロック一共有変数対応情報テーブル（ａ），ロック順序規約記憶手段（ｂ）、共有変数宣言文（ｃ）の構成図である。

【図３】排他制御命令分割手段の処理手順を示すフローチャートである。

【図４】ロック命令移動手段の処理手順を示すフローチャートである。

【図５】アンロック命令移動手段の処理手順を示すフローチャートである。

【図６】従来の排他制御の説明図である。

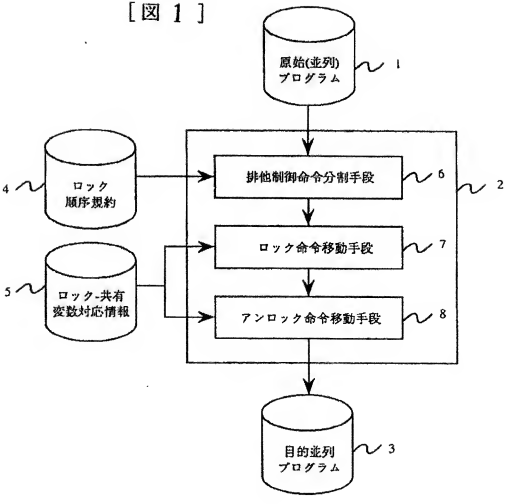
【図７】本発明のロック記述方式と翻訳結果を示す図である。

【符号の説明】

１…原始（並列）プログラム、２…排他制御命令翻訳装置（コンパイラ）、３…目的並列プログラム、４…ロック順序規約記憶手段、５…ロック一共有変数対応情報記憶手段、６…排他制御命令分割手段、７…ロック命令移動手段、８…アンロック命令移動手段。

【図1】

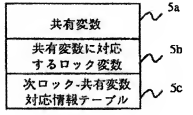
［図 1 ］


原始(並列)
プログラム
1
ロック
順序規約
4
排他制御命令分割手段
6
2
ロック命令移動手段
7
ロック-共有
変数対応情報
5
アンロック命令移動手段
8
目的並列
プログラム
3

【図2】

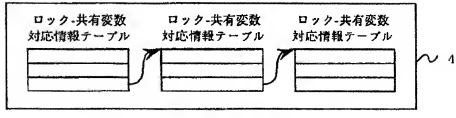
［図２］

(a)



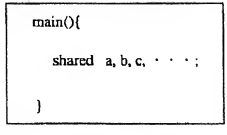
ロック-共有データ対応情報テーブル

(b)



ロック順序規約記憶手段

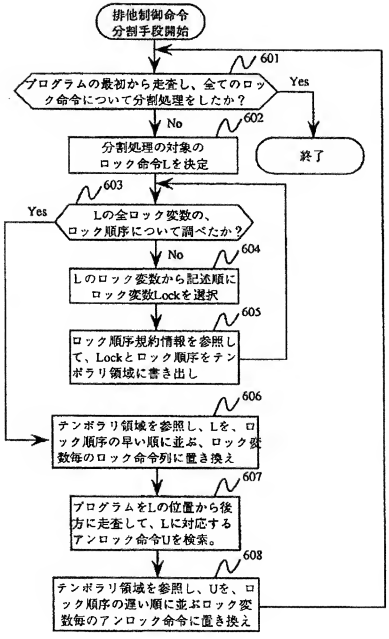
(c)



共有変数の宣言文

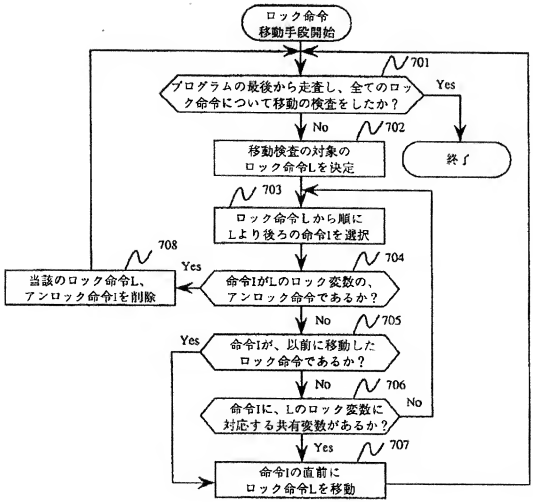
【図3】

［図３］


排他制御命令
分割手段開始
601
プログラムの最初から走査し、全てのロック命令について分割処理をしたか？
Yes
No
602
分割処理の対象のロック命令Lを決定
終了
603
Yes
Lの全ロック変数の、ロック順序について調べたか？
No
604
Lのロック変数から記述順にロック変数Lockを選択
605
ロック順序規約情報を参照して、Lockとロック順序をテンポラリ領域に書き出し
606
テンポラリ領域を参照し、Lを、ロック順序の早い順に並ぶ、ロック変数毎のロック命令列に置き換え
607
プログラムをLの位置から後方に走査して、Lに対応するアンロック命令Uを検索。
608
テンポラリ領域を参照し、Uを、ロック順序の遅い順に並ぶロック変数毎のアンロック命令に置き換え

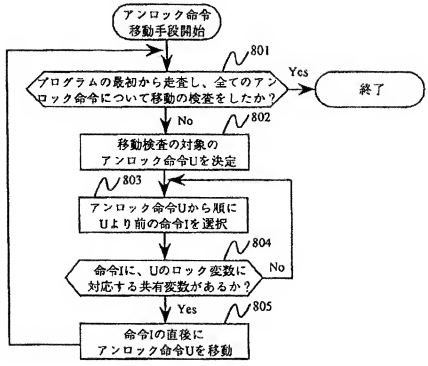
【図4】

［図４］


ロック命令
移動手段開始
701
プログラムの最後から走査し、全てのロック命令について移動の検査をしたか？
Yes
No
702
移動検査の対象のロック命令Lを決定
終了
703
ロック命令Lから順にLより後ろの命令Iを選択
704
命令IがLのロック変数の、アンロック命令であるか？
Yes
708
当該のロック命令L、アンロック命令Iを削除
No
705
命令Iが、以前に移動したロック命令であるか？
Yes
No
706
命令Iに、Lのロック変数に対応する共有変数があるか？
No
Yes
707
命令Iの直前にロック命令Lを移動

【図5】

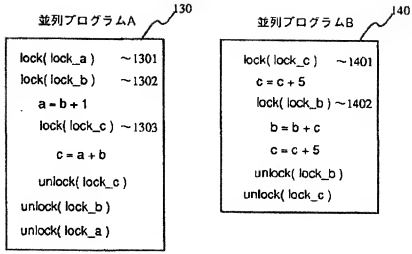
［図５］


アンロック命令
移動手段開始
801
プログラムの最初から走査し、全てのアンロック命令について移動の検査をしたか？
Yes
終了
No
802
移動検査の対象の
アンロック命令Uを決定
803
アンロック命令Uから順に
Uより前の命令Iを選択
804
命令Iに、Uのロック変数に
対応する共有変数があるか？
No
Yes
805
命令Iの直後に
アンロック命令Uを移動

【図６】

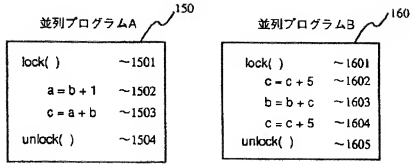
［図６］

(a)



従来のロック記述方式と、その欠点

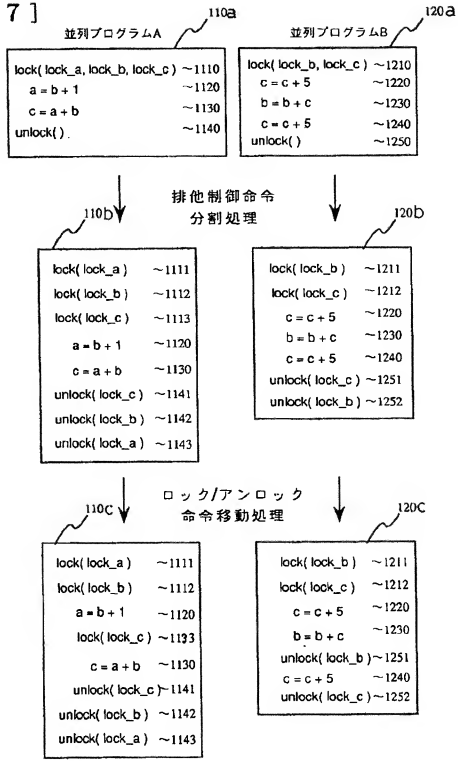
(b)



従来のロック記述方式と、その欠点

【図7】

[図 7]


110a
並列プログラムA
lock(lock_a, lock_b, lock_c) ～1110
a = b + 1 ～1120
c = a + b ～1130
unlock( ) ～1140
120a
並列プログラムB
lock(lock_b, lock_c) ～1210
c = c + 5 ～1220
b = b + c ～1230
c = c + 5 ～1240
unlock( ) ～1250
排他制御命令
分割処理
110b
lock(lock_a) ～1111
lock(lock_b) ～1112
lock(lock_c) ～1113
a = b + 1 ～1120
c = a + b ～1130
unlock(lock_c) ～1141
unlock(lock_b) ～1142
unlock(lock_a) ～1143
120b
lock(lock_b) ～1211
lock(lock_c) ～1212
c = c + 5 ～1220
b = b + c ～1230
c = c + 5 ～1240
unlock(lock_c) ～1251
unlock(lock_b) ～1252
ロック/アンロック
命令移動処理
110c
lock(lock_a) ～1111
lock(lock_b) ～1112
a = b + 1 ～1120
lock(lock_c) ～1113
c = a + b ～1130
unlock(lock_c) ～1141
unlock(lock_b) ～1142
unlock(lock_a) ～1143
120c
lock(lock_b) ～1211
lock(lock_c) ～1212
c = c + 5 ～1220
b = b + c ～1230
unlock(lock_b) ～1251
c = c + 5 ～1240
unlock(lock_c) ～1252